k-Moyennes

N. Gast, E. Gaussier

1 Les k-moyennes

L'algorithme des *k-moyennes*, également appelé algorithme des *nuées dynamiques*, (en anglais *k-means*) est un algorithme couramment utilisé en analyse de données. Il permet de partitionner une collection d'objets en K classes, K étant un nombre fixé par l'utilisateur. On supposera dans la suite que nos objets o_j $(1 \leq j \leq N)$ peuvent être représentés sous forme de vecteurs. L'algorithme des k-moyennes se déroule de la façon suivante :

1. Choisir K objets au hasard parmi les objets de la collection. Soient $(R_1, \cdots, R_K)$ les objets ainsi obtenus. $(R_1, \cdots, R_K)$ sont les représentants de K classes $(C_1, \cdots, C_K)$ qui sont pour l'instant vides.
2. Affecter chaque objet de la collection à l'une des classes en fonction du représentant le plus proche :
$$\text{argmin}_{k, 1 \leq k \leq K}\ d(o_j, R_k)$$
où d est une distance ou une similarité entre objets.
3. Calculer de nouveaux représentants pour les classes. Ces nouveaux représentants correspondent à la moyenne des objets de la classe :
$$\forall k,\ 1 \leq k \leq K,\ R_k = \frac{1}{|C_k|} \sum_{j, o_j \in C_k} o_j$$
4. Retourner en 2 tant que la différence $\Delta(R)$ entre les anciens et les nouveaux représentants est supérieure à un seuil ϵ fixé (et arbitrairement petit).

La complexité de l'algorithme des k-moyennes est de l'ordre de $O(KNIs)$, où I est le nombre d'itérations de l'algorithme et s la complexité du calcul de la distance/similarité. La partition obtenue par l'algorithme des k-moyennes dépend des représentants initialement choisis (essayez de vous en convaincre sur un exemple simple). De façon à s'affranchir en partie de cette dépendance, on exécute l'algorithme des k-moyennes (k et d étant fixés) avec des initialisations différentes, et on retient la meilleure partition. La qualité d'une partition est mesurée par la quantité :

$$D = \sum_{k=1}^{K} \sum_{j, o_j \in C_k} d(o_j, R_k)$$

qui mesure la cohésion des classes obtenues.

Le but de ce TP est de mettre en œuvre cet algorithme sur une collection de documents textuels. Pour cela, nous utiliserons les notations et conventions suivantes :

- $\mathcal{C}$ désigne une collection de N documents et M mots ; d, ou $d_i (1 \leq i \leq N)$, représente un document de la collection et w, ou $w_i (1 \leq i \leq M)$, un mot du vocabulaire.
- Chaque document est représenté par un vecteur de dimension M. La coordonnée du document d sur le ième axe correspond au nombre de fois que le mot w_i apparaît dans d.

– Chaque mot est en fait codé par un entier. Un *fichier dictionnaire* fournit la correspondance entre mots et entiers. Voici par exemple un extrait d'un tel *fichier dictionnaire* :

```
1 abolir
2 tentation
3 dixon
...
```

– Dans la mesure où chaque document ne contient que peu de mots du vocabulaire, on utilise une représentation creuse, de forme :

```
367 23:1 45:12 67:2 88:47
```

qui s'interprète comme "le 367ième document de la collection contient 1 occurrence du 23ième mot du vocabulaire, 12 occurrences du 45ième mot, ...". Dans cette représentation, les indices des mots sont en ordre croissant. Par convention, les mots du vocabulaire qui n'apparaissent pas dans le document (i.e dont le nombre d'occurrences dans le document vaut 0) ne sont pas représentés. Grâce à cette convention, toute l'information est conservée dans la représentation creuse.

– La collection de documents est encodée dans un *fichier collection* dont chaque ligne représente un document, suivant la représentation ci-dessus.

– La mise en œuvre informatique de la représentation creuse d'un document peut se faire à l'aide de deux tableaux (ou un tableau à deux dimensions) qui contiendront respectivement les mots du vocabulaire présents dans le document et le nombre d'occurrences de ces mots. Un document qui contient p mots distincts peut donc être représenté par deux tableaux de taille p (au lieu d'un tableau de taille M si l'on n'utilise pas la représentation creuse; en général, $p << M$). Par exemple, le document précédent (367) peut être représenté par :

0	1	2	3
23	45	67	88
1	12	2	47

Travail demandé

On vous demande d'écrire un programme Java qui prennent en entrée un *fichier dictionnaire* et un *fichier collection* définis comme précédemment et qui réalise l'algorithme des k-moyennes sur ces données. Ce programme sera testé sur deux fichiers qui vous seront fournis.

Il est de plus important, dans le cadre de l'analyse de données, d'apporter le plus grand soin à l'interprétation des résultats et au comportement des modèles en fonction des paramètres retenus. Plusieurs paramètres sont importants dans le cadre des k-moyennes : le nombre K de classes, l'initialisation (aléatoire) des représentants des classes et la fonction de distance/similarité choisie. On vous demande de faire varier ces paramètres et de voir comment varient les résultats. En particulier, on choisira comme fonctions de similarité un simple produit scalaire et un cosinus, et on fera varier le nombre de classes de 10 à 50 avec un pas de 10.